University of Manchester

Deriving Variables From Twitter/X Data: User Guide

Generated for replication as part of the DIGISURVOR project: https://digisurvor.github.io/main/

CITATION: Gaughan, C., Gibson, R., Cantijoch, M., Cernat, A., and Batista-Navarro, R., (2025). *Deriving Variables From Twitter/X Data: User Guide*. Version 1. Manchester: University of Manchester.

Dr. Conor Gaughan, Prof. Rachel Gibson, Dr. Marta Cantijoch, Prof. Alexandru Cernat, Dr. Riza Batista-Navarro

9-19-2025

INTRODUCTION

This user guide was created in conjunction with a working research paper: *"Linking Survey and Social Media Data: Bridging the Gap Between Open Research and Data Protection"*. It is the combined work of Dr. Conor Gaughan, Prof. Rachel Gibson, Dr. Marta Cantijoch, Prof. Alexandru Cernat, and Dr. Riza Batista-Navarro based at the University of Manchester, as part of the UKRI funded 'smart data' project DIGISURVOR. The guiding aim of the project is to investigate the methodological and ethical challenges of linking survey with social media data, and to develop ways in which researchers can better share their social media data with others. Data extracted from social media sites such as Facebook, Twitter (now X), Instagram, and LinkedIn are used frequently in social research, and it is becoming more common for these data to be linked with other data sources such as public opinion surveys. However, due to a range of legal, ethical and platform-specific restrictions, these data cannot be shared openly with other researchers. A particular challenge is that the publicly searchable nature of social media data means that releasing raw textual data risks disclosure of the author's profile, which is especially problematic if linked to confidential survey data. Thus, in lieu of releasing raw social media data, we have established a procedure for the extraction of variables of interest from the data which can describe its structural and semantic content but remains unidentifiable.

The following user guide outlines this process which has been applied specifically to two linked survey-to-Twitter/X datasets of US respondents during the 2020 and 2024 US Presidential election campaigns. In this sense, the guide serves two purposes: 1) to act as a codebook for the two linked datasets which will be of use to researchers wishing to use this data for their own analysis; and 2) to act as a user guide for researchers who wish to replicate this process (or similar) on their own social media datasets. It first outlines the two datasets used as baselines for variable selection and creation including details about why and how they were

collected, their size and features, and some brief details about the linked survey component. The guide then describes the types of variables that have been extracted from this data, followed by descriptions of the required input variables and their data types from the raw data which are subsequently fed into the processing workflow. The guide then details the specific software requirements for running the code scripts which facilitate variable extraction, followed by details of the individual code scripts themselves. The final section provides individual descriptions of each of the 65 and 61 profile-level and tweet-level variables extracted from the raw input data.

TWITTER (X) DATA:

Both datasets were collected for analysis of substantive research questions, as part of independent externally funded projects i.e. not the specific methodological questions posed in this project.[1] They were generated for a European Research Council funded project examining digital campaigning in five electoral democracies and combines two-wave pre- and post-election panel survey responses with participant Twitter/X data. Here, we draw on datasets for two Presidential election time points in the United States (US) – 2020 (1) and 2024 (2). The survey component measures media consumption, perceptions of digital campaign contact, awareness of misinformation, core political attitudes and behaviours, plus standard socio-demographic characteristics and self-reported Twitter/X use.

Wave 1 of Dataset 1 was fielded to 5,952 respondents between 16 September – 20 October 2020, and Wave 2 was fielded in the week following the election (9 November). Respondents to the pre-election survey were invited to share their Twitter/X handle with the research team. 2,460 participants reported having an active Twitter/X account (41%) and 1,598 agreed to share their handle (27% of overall sample; 65% of those with an account). There were four stages of attrition: 190 did not subsequently provide a handle, 171 provided an empty handle, and of the remaining 1,237, 920 could be validated against the Twitter/X API, which constituted 15% of the overall sample. Dataset 1 contains the collected timelines of these respondents during the period 1 September 2020 – 9 November 2020. Of the 920 respondents, we were able to extract profile-level variables for 905 and post-level variables for 539 of them respectively. Between the 539 respondents who had tweeted, there were 222,365 posts in total.

[1] Both were originally collected by Prof. Rachel Gibson for the European Research Council funded Project 833177(2020- 2025) Digital Campaigning and Electoral Democracy (DiCED).

Wave 1 of Dataset 2 was fielded to 5,757 respondents between 24th September and 21st October 2024 and Wave 2 was fielded in the two weeks following the election (12th-26th November 2024). Similar attrition rates were recorded in this survey: 2,621 had an active Twitter/X account (46%) and 1,570 agreed to share their handle (27% of overall sample; 60% of those with an account). There were four stages of attrition: 391 did not subsequently provide a handle and of the remaining 1,179, 963 could be validated against the Twitter/X API, which constituted 17% of the overall sample. As with Dataset 1, we collected the timelines of these respondents during the period 10th September – 4th November**.** We extracted profile-level variables for all 963 respondents, and post-level variables for 375. Between the 375 that had tweeted, there were 46,232 posts.

VARIABLE CREATION:

We divide the SMD into two units of analysis: (1) Profile-Level; (2) Post-Level. We then divide the variables we want to derive from these units into two types: **Structural** and **Semantic**. We define structural variables as those which broadly *describe* the general content and structure of a respondent's digital data. This can include descriptive metrics such as a respondent's length of membership on the platform, their total number of followers, the frequency of their posting, their overall use of attachments and URLs in their posts, counting the number of a characters in the text of a post, or the average amount of engagement a post receives. We define semantic variables as more computationally complex variables such as topic classification, lexical grammaticality and sophistication, language readability, toxicity and sentiment analysis, and ideological bias detection.

Variable Level	**Overall**
Profile-Level	65
Post-Level	61
Overall	126

To create these variables, we enlist several out-of-the-box computational tools which seek to combine optimal reliability with maximum simplicity and replicability. Thus, we do not train any models of our own or build any new variables that rely exclusively on the data we have. Moreover, the variables we create are not designed to be completely prescriptive or exhaustive. Instead, the aim here is to build a transparent and replicable *template* which can provide the skeleton code for others to implement and build upon using their own SMD. Additionally, we acknowledge that due to the authors' domain specific knowledge as well as the inherently political nature of the datasets we are using, some of these variables may be slanted towards a more political science audience. However, we believe that most of these baseline variables are applicable to many fields, but we encourage researchers to tailor these variables or generate

new ones according to their specific data and research objectives. The datasets we are using are also primarily in English given that we are using a US dataset and so certain variables may have to be adjusted for use on non-English datasets. All variables are built using the programming language *Python* (*v.3.11.7)*.

Metadata Variables

These are perhaps the most straightforward variables we can extract from any SMD. In most cases, this involves simple count measures, categorical variables or Booleans to describe the general structure of a unit of SMD. For (our) post-level data, this includes categorising the post type (original, retweet, reply, quote), the date it was created (timestamp is truncated to avoid too much granularity), engagement metrics (like count, retweet count, reply count, quote count), Booleans for whether it contains any media attachments or geotags, and the post reply settings. At the user profile level, we generate similar metadata metrics such as profile created at date, profile metrics (follower count, following count, like count, post count, listed count), as well as Booleans for whether additional profile fields are populated (display name, location, description). Where these fields are populated, we extract additional variables including matching listed locations to official geo-location data using the Nominatim API accessed via Python's *geopy* module as well as cross-referencing display names with real first and last names using Python's *names-dataset* module.

Text Variables

Practically all SMD contains text-based fields. Text is the lifeblood of social networking, microblogging or forum platforms such as Facebook, Twitter/X, Reddit and LinkedIn. For media-based platforms such as YouTube, Instagram, TikTok and Snapchat, multimodal data like images, videos and audio are much more central, though text is still an important aspect of these SMD. Due to its greater complexity and to the fact we are using Twitter/X data, we do

not focus on multimodal data in this paper (though we do believe this is an important area to expand on in future work). Text is perhaps the most identifiable unit of information within our SMD which cannot be released in its raw format without significant risk of disclosure. Thus, we derive an array of anonymised structural and semantic variables from the text at both the post and user level.

First, we provide a set of variables which describe the lexical composition of the text. This involves simple count metrics for the total number of characters, number of textual characters and the number of capital letters, as well as the number of letters, words, unique words, sentences, syllables, monosyllables, and polysyllables. This can provide researchers with valuable information about the lexical and syntactic complexity of a text. Additionally, we derive a grammaticality score based on the Python's *LanguageTool* AI Grammer Checker, as well as a set of readability scores (Flesch Reading Ease, McAlpine EFLAW Score, Reading Grade) derived using the *textstat* module. We also calculate count measures for the number of special characters that appear in a text including the number of exclamation marks, question marks, emojis, @mentions, hashtags, and URLs. We believe these variables can be applied almost universally to any SMD text at both the post and user profile level, though one may wish to tailor it to capture specific nuances depending on the data, language, and platform.

URL Variables

Many SMD text contain URLs which are an important aspect of content sharing and information dissemination, especially on microblogging sites like Twitter/X. Thus, not only do we count the number of URLs that appears in a body of text, we also match the domain level of the URL to the Media Bias/Fact Check (MBFC) media bias website.[2] MBFC is a comprehensive resource which contains media bias and credibility scores for thousands of

[2] The Media Bias/Fact Check website can be found here: https://mediabiasfactcheck.com/

major media sources, journalists and politicians and can be a useful tool for assessing the ideological slant and factual credibility of the information that users share. It is an imperfect method as it will not contain scores for all URLs if the domain does not match to anything in the MBFC dataset (typically because it is not a news source or it is too local). However, where users have shared information from relatively sized news sources, we can derive useful information about ideological balance and quality of content sharing, as well as content engagement (where users have engaged with posts that contain URLs).

TweetNLP

Finally, the following semantic variables are generated using the Python module *TweetNLP*.[3] TweetNLP is a free-to-use module which supports a suite of natural language processing (NLP) tasks fine-tuned to SMD text. The NLP tasks supported include sentiment analysis, emotion recognition, offensive language identification, topic classification, irony detection, and named entity recognition. These are all implemented out-of-the-box in Python to derive semantic variables from the SMD text to give researchers a broad level understanding of the kinds of things users talk about and how they talk about it.

[3] More details can be found here: https://github.com/cardiffnlp/tweetnlp

INPUT VARIABLES:

Input columns must be named as follows and in the correct data type. The code currently accepts .pkl files to ensure that data types are maintained.

PROFILE DATA:

Variable Name	Type	Description
ID	String	The unique user ID
created_at	String	Date and time the profile was created
followers_count	Integer	Number of account followers
following_count	Integer	Number of accounts followed
tweet_count	Integer	Number of tweets posted
listed_count	Integer	Number of lists the account appears on
display_name	String	Profile display name
screen_name	String	Profile screen name (username/handle)
location	String	Profile location field
description	String	Profile description field (bio)

TWEET DATA:

Variable Name	Type	Description
ID	String	The unique user ID
tweet_id	String	The unique tweet ID
referenced_tweet_type	String	Type of tweet
created_at_x	String	Date and time the author's tweet was created
source_x	String	Source of posting (iPhone, Browser, etc.)
language_x	String	Language of post text
retweet_count_x	Integer	Number of author retweets
reply_count_x	Integer	Number of author replies
like_count_x	Integer	Number of author likes
quote_count_x	Integer	Number of author quotes
media_keys	String	String of unique keys for media attachments within the text (GIFs, Videos, Images etc,)
geo_coordinates	String	String of geo-coordinates (Long, Lat)
reply_settings_x	String	Author reply settings
created_at_y	String	Date and time the original tweet was created
source_y	String	Original source of posting (iPhone, Browser, etc.)
language_y	String	Original language of post text
retweet_count_y	Integer	Number of original author retweets
reply_count_y	Integer	Number of original author replies
like_count_y	Integer	Number of original author likes
quote_count_y	Integer	Number of original author quotes
tweet_text	String	Text of the tweet
entities_urls	String	The entities returned by the API. This parsed to extract the expanded URL.

SOFTWARE REQUIREMENTS

All output variables were extracted using the programming language Python (v.3.11.7). The following modules are required:

Natural Language Processing (NLP)		
re	Regular expression operations	https://docs.python.org/3/library/re.html
textstat	Calculate statistics from text	https://textstat.org/
emoji	Analyse emojis in text	https://pypi.org/project/emoji/
language_tool_python	Wrapper for LanguageTool: used for grammar checking	https://pypi.org/project/language-tool-python/
tweetnlp	Collection of NLP models for analysing tweet text	https://github.com/cardiffnlp/tweetnlp
rapidfuzz	Fuzzy string matching	https://rapidfuzz.github.io/RapidFuzz/
nameparser (HumanName)	Parsing human names into their individual components	https://pypi.org/project/nameparser/
tldextract	Parsing URLs strings	https://pypi.org/project/tldextract/
Language, Name and Location Detection		
langdetect	Detecting the language of text	https://pypi.org/project/langdetect/
geopy (Nominatim)	Python client for geocoding web services	https://geopy.readthedocs.io/en/stable/
names_dataset	First and last name matching	https://pypi.org/project/names-dataset/
Data Wrangling		
pandas	Fast and flexible data wrangling	https://pypi.org/project/pandas/
System and Utility		
time	Time access and conversions	https://docs.python.org/3/library/time.html
random	Generate pseudo-random numbers	https://docs.python.org/3/library/random.html
argparse	Parse command line arguments	https://docs.python.org/3/library/argparse.html
multiprocessing	Process-based parallelism	https://docs.python.org/3/library/multiprocessing.html
tqdm	Process bar	https://pypi.org/project/tqdm/2.2.3/

PROCESS EXECUTION

There are eight code scripts in total that comprise the variable extraction process which can be executed via the command line using the following script:

derive_social_media_variables.py

with the accompanying arguments:

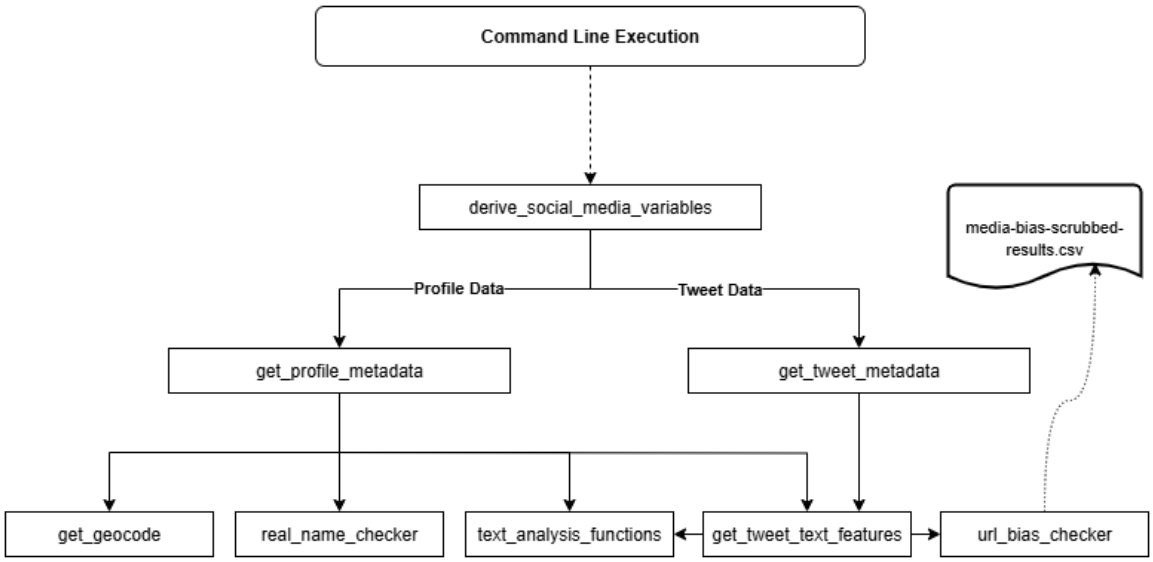
Argument	Description	Type	Required?	Options	Default
input	Path to input file	String	Yes	N/A	N/A
type	Input data type	String	Yes	‘profile’, ‘tweet’	N/A
file_path	Output file path (no extension!)	String	Yes	N/A	N/A
batch_size	Number of rows per batch	Integer	No	N/A	10
start_row	Row number to begin from	Integer	No	N/A	0
nproc	Number of processors to use	Integer	No	N/A	None
verbose	Enable verbose output	Store True	No	N/A	Defaults to True if argument is used

By default, the input file (profile_data.pkl or tweet_data.pkl), the data type (profile or tweet) and an output file path (automatically suffixed with “_rows_X_to_Y.csv”) are all required. Be default, the input data provided is split into batches of 10 rows each for efficiency, but this can be optionally set higher or lower. Start row defaults to 0 but may be useful if the process breaks midway through after some rows have already been processed. The process itself can be relatively slow and process-based parallelism is relied upon to speed up the procedure, where batches of rows are processed in parallel to one another. The more processors that are used, the more rows that can be processed at once, although we recommend using less than the total number of processors available on the machine at once. If number of processors is not specified, it will default to the total number of processors on the machine minus one. If the verbose argument is specified, this will generate a process bar to track progress.

Example:

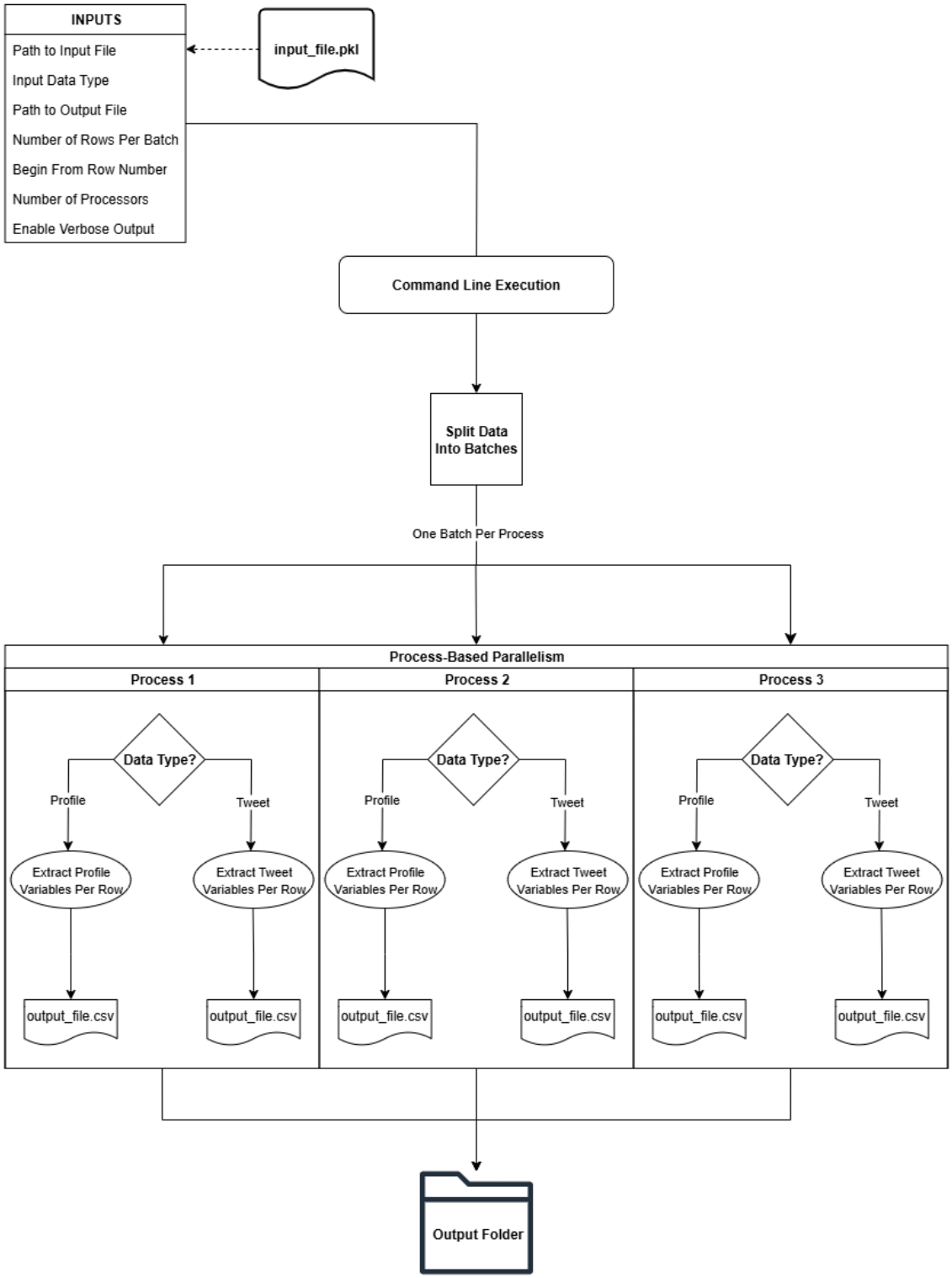
python.exe derive_social_media_variables.py --input input_file.pkl --type tweet --file_path tweet_variables --batch_size 1000 --start_row 0 --nproc 6 --verbose

CODE STRUCTURE



The process relies on a total of eight unique code scripts. The derive_social_media_variables script is responsible for ingesting the input data file, splitting it into the required number of batches based on the specified batch size, and then executing each batch file in parallel via a designated processor. Where the number of overall batches exceeds the overall number of processors, these are queued until a previous batch is complete and a processor is freed. Depending on the data type, each processor will execute the **get_profile_metadata** or **get_tweet_metadata** scripts to extract new variables from the input data which is subsequently saved out as a .csv file to the designated file path. Ensure that all code scripts (and the accompanying media-bias .csv file) are all stored in the same directory (folder) along with an empty Python file titled __init__.py. This will mark the directory as a module and allow Python to import functions from the other scripts.

DATA WORKFLOW:



OUTPUT VARIABLES

User-Level Variables

Variable Name	**Description**	**Type**	**Software Dependency (Python Module)**	**Software Dependency (Module Function)**
ID	A unique pseudonymised ID for the respondent	String		
Profile Metrics: These variables are returned directly by the platform API				
created_date	Date the account was created	String	datetime	date().strftime()
followers_count	Total number of followers	Integer		
following_count	Total number of accounts following	Integer		
tweet_count	Total number of tweets posted	Integer		
listed_count	Total number of lists the account appears on	Integer		
Display Name (DN): These variables are derived from the account display name field				
DN_length	Total number of characters	Integer	built-in	len()
The following variables were applied to pre-processed text. This involves the removal of URLs, emojis, hashtags, @mentions and extra whitespace. This is conducted using a custom-built strip_text() function which relies on the *re* and *emoji* modules. (Excl. the hashtag and emoji count variables)				
DN_character_length	Total number of textual characters	Integer	textstat	char_count()
DN_capt_character_count	Total number of capitalised characters	Integer	built-in	isupper()
DN_capt_character_prop	Proportion of capitalised characters to total number of characters	Float		
DN_letter_count	Total number of letters	Integer	textstat	letter_count()
DN_word_count	Total number of words	Integer	textstat	lexicon_count()

DN_unique_word_count	Total number of unique words	Integer	built-in	set()
DN_exclamation_count	Total number of exclamation marks	Integer	built-in	count()
DN_question_mark_count	Total number of question marks	Integer	built-in	count()
DN_hashtag_count	Total number of hashtags	Integer	re	findall()
DN_emoji_count	Total number of emojis	Integer	emoji	emoji_count()
For human name handling, a selection of functions were used. The HumanName() function from the *nameparser* module was used to parse listed names and salutations and first and last names were matched against the NameDataset from the *names_dataset* module: https://pypi.org/project/names-dataset/. The string similarity between a user's display name and their handle (username) is the Levenshtein Distance similarity ratio computed using the ratio() function from the *rapidfuzz* module.				
DN_handle_similarity	Levenshtein Distance similarity between display name and handle (username)	Float	from rapidfuzz import fuzz	ratio()
DN_listed_salutation	Did the user list a salutation?	Boolean	from nameparser import HumanName	HumanName().title
DN_used_first_name	Did the user list a real first name?	Boolean	from names_dataset import NameDataset	search()
DN_used_last_name	Did the user list a real last name?	Boolean	from names_dataset import NameDataset	search()
Location (LC): These variables are derived from the account location field				
LC_field_populated	Was the location field populated?	Boolean		
LC_level_count	Total number of location levels listed. This is ascertained by number of words separated by a comma or forward slash.	Integer	re	split(r'[,/]')
Geo-location matching is conducted through the Nominatim API which uses OpenStreetMap data for geocoding. It accepts free-form search queries and is accessed via the *geopy.geocoders* module.				

LC_levels_identified	Total number of levels that can be matched to an official geographical location	Integer	from geopy.geocoders import nominatim	geocode()
LC_level_types	Location types identified	List of Strings	from geopy.geocoders import nominatim	geocode().raw.get (“addresstype)
Description (DS): These variables are derived from the account description field				
DS_field_populated	Was the description field populated?	Boolean		
DS_language	Predicted text language	String	lang_detect	detect()
DS_post_length	Total number of characters	Integer	built-in	len()
The following variables were applied to pre-processed text. This involves the removal of URLs, emojis, hashtags, @mentions and extra whitespace. This is conducted using a custom-built strip_text() function which relies on the *re* and *emoji* modules. (Excl. the hashtag, mention and emoji count variables)				
DS_character_length	Total number of textual characters	Integer	textstat	char_count()
DS_capt_character_count	Total number of capitalised characters	Integer	built-in	is.upper()
DS_capt_character_prop	Proportion of capitalised characters to total number of characters	Float		
DS_letter_count	Total number of letters	Integer	textstat	letter_count()
DS_word_count	Total number of words	Integer	textstat	lexicon_count()
DS_unique_word_count	Total number of unique words	Integer	built-in	set()
DS_sentence_count	Total number of sentences	Integer	textstat	sentence_count()
DS_syllable_count	Total number of syllables	Integer	textstat	syllable_count()
DS_monosyllable_count	Total number of monosyllables	Integer	textstat	monosyllabcount()
DS_polysyllable_count	Total number of polysyllables	Integer	textstat	polysyllabcount()
DS_grammar_score	Proportion of grammatical errors to total number of words	Float	language_tool_python	tool_check()
DS_reading_ease	Flesch Reading Ease Score. The higher the score, the easier the readability of the text. (Max 121.22)	Float	textstat	flesch_reading_ease()

DS_non_eng_reading_ease	McAlpine EFLAW Readability Score. Readability of an English text for a foreign learner of English. It is recommended to aim for a score equal to or lower than 25	Float	textstat	mcalpine_eflaw()
DS_reading_grade	Estimated school grade level required to understand the text. Consensus score based on seven separate readability tests	Float	textstat	text_standard()
DS_exclamation_count	Total number of exclamation marks	Integer	built-in	count()
DS_question_mark_count	Total number of question marks	Integer	built-in	count()
DS_hashtag_count	Total number of hashtags	Integer	re	findall()
DS_emoji_count	Total number of emojis	Integer	emoji	emoji_count()
DS_mentions_count	Total number of @mentions	Integer	re	findall()
DS_url_count	Total number of URLs	Integer	re	findall()
Twitter/X reformats in-text URLs to its own domain: https://t.co/. To find the original URL domain, the full URL listed in the extended_urls column must be used. The domain of URL is extracted using the extract().domain function from the *tldextract* module and matched to a list of domains in a pre-built CSV file scrubbed from the Media Bias/Fact Check website: https://mediabiasfactcheck.com/. This contains 1579 unique domains for major media sources around the world, as well as scores of their ideological bias and reporting credibility.				
DS_url_mbfc_matches	Total number of URL domain matches against the Media Bias/Fact Check (MB/FC) database	Integer	built-in	Isin()
DS_url_mbfc_bias	Media Bias/Fact Check bias score(s) for URL domain matches	Integer		
DS_url_mbfc_credibility	Media Bias/Fact Check factual reporting credibility score(s) for URL domain matches	Integer		
The following variables are estimated using pre-trained models from the *TweetNLP* module*:* https://github.com/cardiffnlp/tweetnlp				
DS_topics	Topic classification label(s)	String	tweetnlp.load_model('topic_classification')	topic()
DS_topic_prob	Certainty estimates for each topic label	Float		

DS_sentiment	Sentiment classification label	String	tweetnlp.load_model('sentiment')	sentiment()
DS_sentiment_prob	Certainty estimate for the sentiment classification	Float		
DS_irony	Irony classification label	String	tweetnlp.load_model('irony')	irony()
DS_irony_prob	Certainty estimate for the irony classification	Float		
DS_offensive	Offensive language classification label	String	tweetnlp.load_model('offensive')	offensive()
DS_offensive_prob	Certainty estimate for the offensive language classification	Float		
DS_emotion	Emotion classification label	String	tweetnlp.load_model('emotion')	emotion()
DS_emotion_prob	Certainty estimate for the emotion classification	Float		
DS_hate	Hate speech classification label	String	tweetnlp.load_model('hate')	hate()
DS_hate_prob	Certainty estimate for the hate speech classification	Float		
DS_entity_count	Total number of named entities recognised	Integer	tweetnlp.load_model('ner')	ner()
DS_entity_types	Types of named entities recognised	String		
DS_entity_prob	Certainty estimates for each recognised entity	Float		

Post-Level Variables

Variable Name	**Description**	**Type**	**Software Dependency (Python Module)**	**Software Dependency (Module Function)**
ID	A unique pseudonymised ID for the respondent who authored the post	String		
post_type	Is it an original authored tweet, retweet, quote tweet or a reply?	String		
Author Tweet: This is metadata about the *author's* tweet (respondent).				
created_date	Date the author's tweet was posted	String	datetime	date().strftime()
post_source	Digital source of author's posting	String		
post_language	Language of the author's post	String		
retweet_count	Total number of times the author's post was retweeted	Integer		
reply_count	Total number of times the author's post was replied to	Integer		
like_count	Total number of times the author's post was liked/favourited	Integer		
quote_count	Total number of times the author's post was quote tweeted	Integer		
total_engagement_count	Total amount of engagement the author's post received	Integer		
reply_ratio	Total number of replies divided by the total number of retweets and likes	Integer		
contains_media	Does the post contain a media attachment (Image/Video/GIF)?	Boolean		
contains_geotag	Does the post contain a geotag?	Boolean		
reply_settings	What reply settings did the author set for the post?	String		
Tweet Text: These variables are derived from the post text field				

TEXT_post_length	Total number of characters	Integer	built-in	len()
The following variables were applied to pre-processed text. This involves the removal of URLs, emojis, hashtags, @mentions and extra whitespace. This is conducted using a custom-built strip_text() function which relies on the *re* and *emoji* modules. (Excl. the hashtag, mention and emoji count variables)				
TEXT_character_length	Total number of textual characters	Integer	textstat	char_count()
TEXT_capt_character_count	Total number of capitalised characters	Integer	built-in	is.upper()
TEXT_capt_character_prop	Proportion of capitalised characters to total number of characters	Float		
TEXT_letter_count	Total number of letters	Integer	textstat	letter_count()
TEXT_word_count	Total number of words	Integer	textstat	lexicon_count()
TEXT_unique_word_count	Total number of unique words	Integer	built-in	set()
TEXT_sentence_count	Total number of sentences	Integer	textstat	sentence_count()
TEXT_syllable_count	Total number of syllables	Integer	textstat	syllable_count()
TEXT_monosyllable_count	Total number of monosyllables	Integer	textstat	monosyllabcount()
TEXT_polysyllable_count	Total number of polysyllables	Integer	textstat	polysyllabcount()
TEXT_grammar_score	Proportion of grammatical errors to total number of words	Float	language_tool_python	tool_check()
TEXT_reading_ease	Flesch Reading Ease Score. The higher the score, the easier the readability of the text. (Max 121.22)	Float	textstat	flesch_reading_ease()
TEXT_non_eng_reading_ease	McAlpine EFLAW Readability Score. Readability of an English text for a foreign learner of English. It is recommended to aim for a score equal to or lower than 25	Float	textstat	mcalpine_eflaw()
TEXT_reading_grade	Estimated school grade level required to understand the text. Consensus score based on seven separate readability tests	Float	textstat	text_standard()
TEXT_exclamation_count	Total number of exclamation marks	Integer	built-in	count()
TEXT_question_mark_count	Total number of question marks	Integer	built-in	count()
TEXT_hashtag_count	Total number of hashtags	Integer	re	findall()
TEXT_emoji_count	Total number of emojis	Integer	emoji	emoji_count()

TEXT_mentions_count	Total number of @mentions	Integer	re	findall()
TEXT_url_count	Total number of URLs	Integer	re.findall()	
Twitter/X reformats in-text URLs to its own domain: https://t.co/. To find the original URL domain, the full URL listed in the extended_urls column must be used. The domain of URL is extracted using the extract().domain function from the *tldextract* module and matched to a list of domains in a pre-built CSV file scrubbed from the Media Bias/Fact Check website: https://mediabiasfactcheck.com/. This contains 1579 unique domains for major media sources around the world, as well as scores of their ideological bias and reporting credibility.				
TEXT_url_mbfc_matches	Total number of URL domain matches against the Media Bias/Fact Check (MB/FC) database	Integer	built-in	Isin()
TEXT_url_mbfc_bias	Media Bias/Fact Check bias score(s) for URL domain matches	Integer		
TEXT_url_mbfc_credibility	Media Bias/Fact Check factual reporting credibility score(s) for URL domain matches	Integer		
The following variables are estimated using pre-trained models from the *TweetNLP* module*:* https://github.com/cardiffnlp/tweetnlp				
TEXT_topics	Topic classification label(s)	String	tweetnlp.load_model('topic_classification')	topic()
TEXT _topic_prob	Certainty estimates for each topic label	Float		
TEXT _sentiment	Sentiment classification label	String	tweetnlp.load_model('sentiment')	sentiment()
TEXT _sentiment_prob	Certainty estimate for the sentiment classification	Float		
TEXT _irony	Irony classification label	String	tweetnlp.load_model('irony')	irony()
TEXT _irony_prob	Certainty estimate for the irony classification	Float		
TEXT _offensive	Offensive language classification label	String	tweetnlp.load_model('offensive')	offensive()
TEXT _offensive_prob	Certainty estimate for the offensive language classification	Float		

TEXT _emotion	Emotion classification label	String	tweetnlp.load_model('emotion')	emotion()
TEXT _emotion_prob	Certainty estimate for the emotion classification	Float		
TEXT _hate	Hate speech classification label	String	tweetnlp.load_model('hate')	hate()
TEXT _hate_prob	Certainty estimate for the hate speech classification	Float		
TEXT _entity_count	Total number of named entities recognised	Integer	tweetnlp.load_model('ner')	ner()
Referenced Tweet: (RT) – If the tweet is a retweet, quote tweet, or a reply, the API also returns information about the original tweet. This is metadata about the *original* (referenced) tweet.				
RT_referenced_author_ID	Pseudonymised ID of the author of the original tweet	String		
RT_created_date	Date the original tweet was posted	String		
RT_post_source	Digital source of original posting	String		
RT_post_language	Language of the original post	String		
RT_retweet_count	Total number of times the original post was retweeted	Integer		
RT_reply_count	Total number of times the original post was replied to	Integer		
RT_like_count	Total number of times the original post was liked/favourited	Integer		
RT_quote_count	Total number of times the original post was quote tweeted	Integer		
RT_total_engagement_count	Total amount of engagement the original post received	Integer		
RT_reply_ratio	Total number of replies divided by the total number of retweets and likes	Integer		
RT_contains_media	Does the original post contain a media attachment (Image/Video/GIF)?	Boolean		
RT_contains_geotag	Does the original post contain a geotag?	Boolean		